Connect Four

Boca Paul-Adrian

Introduction

Connect Four is a two-player connection game in which the player first is given a color by the server and then take turns dropping one colored disc from the top into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. Connect Four is a solved game. The first player can always win by playing the right moves.

Used technologies

The main technology used is TCP. The reasons for using TCP are that:

- TCP numbers the packets received from the server so the recipient can get them in order.
- TCP is serious about reliability. The packets are checked for errors to make sure the request is fulfilled correctly.
- TCP packets are tracked to make sure that no data is lost in between.
- Packets are also checked for corruption.

A UDP connection might be faster but there is a chance that some packets won't be received by the client. In this game we need a TCP connection because it's reliable and we can be sure that a player's move won't go missing and there will be no errors concerning this matter.

Application architecture

The two players will be connected to the server, then the server will assign a different symbol for both of them and will decide who will start.

In a round each player will receive the game matrix from the server, put the corresponding piece in the matrix, check if they've won and then send to the server the matrix and the information regarding the winner (if they have won or not).

The server will send to the second player the updated matrix and the second player will do as the first.

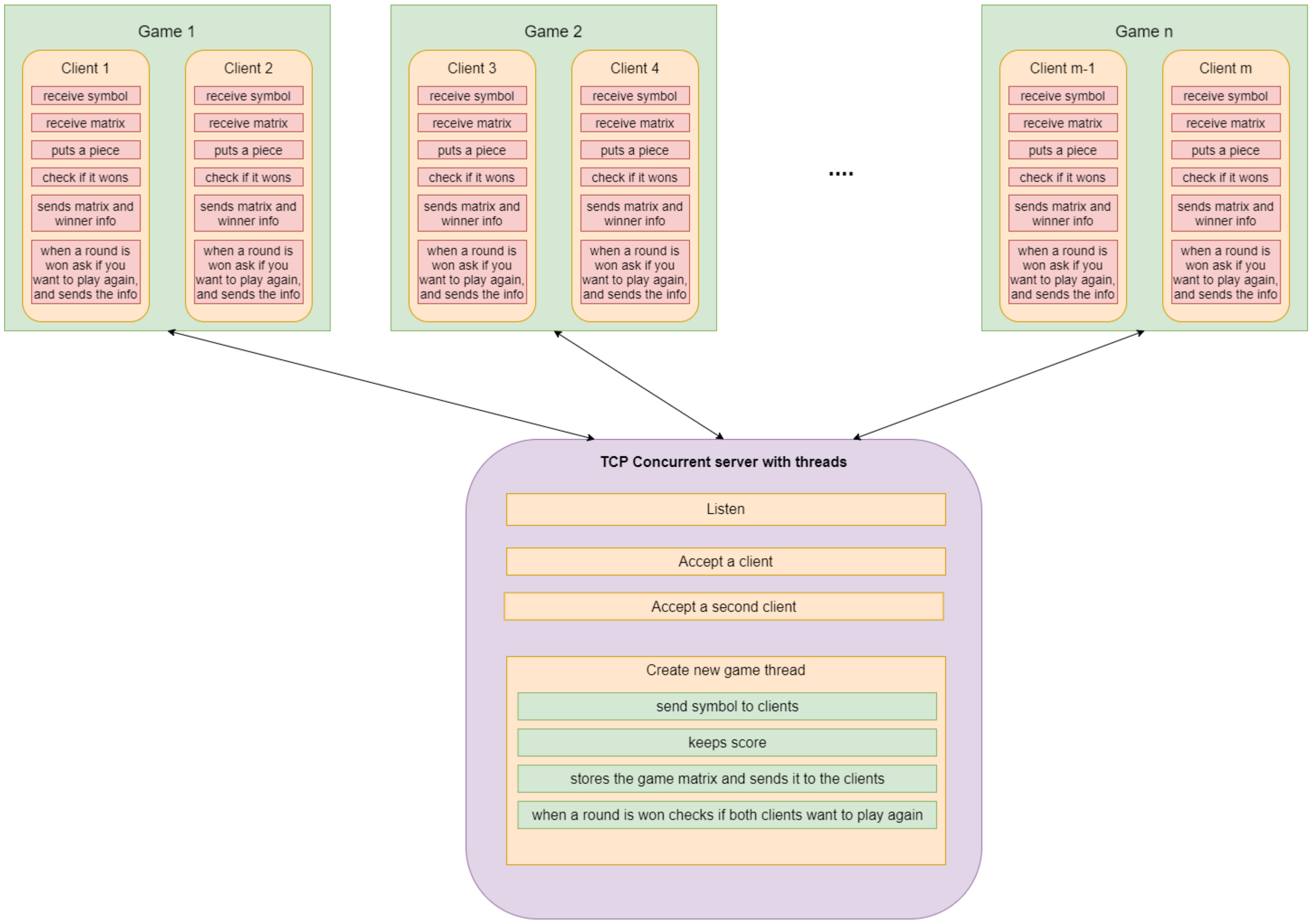
The scenario will repeat itself until one of the player wins.

The server will also keep the score and will show it at the end of each round.

When a round is ended both players are asked if they want to play again.

The server will process they're answers and if both of them are *yes*, then they will play another round.

The game ends when either of them says *no* at the end of a round.



Game 1
Client 1
receive symbol
receive matrix
puts a piece
check if it wons
sends matrix and winner info
when a round is won ask if you want to play again, and sends the info
Client 2
Game 2
Client 3
Client 4
....
Game n
Client m-1
Client m
TCP Concurrent server with threads
Listen
Accept a client
Accept a second client
Create new game thread
send symbol to clients
keeps score
stores the game matrix and sends it to the clients
when a round is won checks if both clients want to play again

Figure 1: Application diagram

Implementation details

The loop in witch we wait for clients to connect and create a new thread when 2 of them are connected:

```
while (1)
   {
      printf ("[server]Asteptam la adresa %s si la portul %d...\n", address, PORT);
      fflush (stdout);
      int cllen = sizeof (client);
      int clsd = accept (sd, (struct sockaddr *) &client, (socklen_t*) &cllen);
      if (clsd < 0)
      {
         perror ("[server]Eroare la accept().\n");
         continue;
      }

      int cllen2 = sizeof (client2);
      int clsd2 = accept (sd, (struct sockaddr *) &client2, (socklen_t*) &cllen2);
      if (clsd2 < 0)
      {
         perror ("[server]Eroare la accept2().\n");
```

```
        continue;
    }

    thData * td;

    td=(struct thData*)malloc(sizeof(struct thData));
    td->idThread=i++;
    td->cl1=clsd;
    td->cl2=clsd2;

    pthread_create(&th[i], NULL, &treat, td);
}
```

The part of the server in witch we manage the comunication with the first player:

```
sendMatrix(tdL.cl1, tabla);
        sendLib(tdL.cl1, lib);
        getMatrix(tdL.cl1, tabla);
        getLib(tdL.cl1, lib);

        send (tdL.cl1, &winner, 1, 0 );
        recv (tdL.cl1, &winner, 1, 0 );

        if (winner == 'X') {
            scor_x ++;
        }

        char_scor_x = char(scor_x);
        char_scor_0 = char(scor_0);

        send (tdL.cl1, &char_scor_x, 1, 0 );
        send (tdL.cl1, &char_scor_0, 1, 0 );
```

The part of the server in witch we manage the comunication with the second player:

```
sendMatrix(tdL.cl2, tabla);
        sendLib(tdL.cl2, lib);
        getMatrix(tdL.cl2, tabla);
        getLib(tdL.cl2, lib);

        send (tdL.cl2, &winner, 1, 0 );
        recv (tdL.cl2, &winner, 1, 0 );

        if (winner == '0') {
            scor_0 ++;
        }

        char_scor_x = char(scor_x);
        char_scor_0 = char(scor_0);

        send (tdL.cl2, &char_scor_x, 1, 0 );
        send (tdL.cl2, &char_scor_0, 1, 0 );
```

The part of the server in witch we process the answers at the end of a round:

```
recv (tdL.cl1, &confirmare1, 1, 0 );
    recv (tdL.cl2, &confirmare2, 1, 0 );

    send (tdL.cl1, &confirmare2, 1, 0 );
    send (tdL.cl2, &confirmare1, 1, 0 );
    if ((strcmp(confirmare1, "y") == 0 || strcmp(confirmare1, "Y") == 0) &&
        (strcmp(confirmare2, "y") == 0 || strcmp(confirmare2, "Y") == 0)) {
        won = 0;
    } else {
        run = 0;
    }
```

Conclusions

The “Connect Four” game is a 2 players game which can be properly implemented using a TCP connection because even though a TCP can be slower it’s also more reliable and for this game we need reliability more than speed.

Bibliography

https://en.wikipedia.org/wiki/Connect_Four
https://www.vpnmentor.com/blog/tcp-vs-udp/